W3School AngularJS 教程

wizardforcel

Published with GitBook



目錄

W3School AngularJS教程

来源：AngularJS 教程

整理：飞龙

AngularJS 简介

AngularJS 是一个 **JavaScript** 框架。它可通过 <script> 标签添加到 HTML 页面。

AngularJS 通过 指令 扩展了 HTML，且通过 表达式 绑定数据到 HTML。

AngularJS 是一个 JavaScript 框架

AngularJS 是一个 JavaScript 框架。它是一个以 JavaScript 编写的库。

AngularJS 是以一个 JavaScript 文件形式发布的，可通过 script 标签添加到网页中：

```
<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'
```



> 我们建议把脚本放在 <body> 元素的底部。
> 这会提高网页加载速度，因为 HTML 加载不受制于脚本加载

AngularJS 扩展了 HTML

AngularJS 通过 **ng-directives** 扩展了 HTML。

ng-app 指令定义一个 AngularJS 应用程序。

ng-model 指令把元素值（比如输入域的值）绑定到应用程序。

ng-bind 指令把应用程序数据绑定到 HTML 视图。

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="">
  <p>在输入框中尝试输入：</p>
  <p>姓名：<input type="text" ng-model="name"></p>
  <p ng-bind="name"></p>
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

</body>
</html>
```

实例讲解：

当网页加载完毕，AngularJS 自动开启。

ng-app 指令告诉 AngularJS，<div> 元素是 AngularJS 应用程序 的"所有者"。

ng-model 指令把输入域的值绑定到应用程序变量 **name**。

ng-bind 指令把应用程序变量 name 绑定到某个段落的 innerHTML。

>  如果您移除了 **ng-app** 指令，HTML 将直接把表达式显示出来，不会去计算表达式的结果。

什么是 AngularJS？

"AngularJS 是专门为应用程序设计的 HTML。"

AngularJS 使得开发现代的单一页面应用程序（SPAs：Single Page Applications）变得更加容易。

- AngularJS 把应用程序数据绑定到 HTML 元素。
- AngularJS 可以克隆和重复 HTML 元素。
- AngularJS 可以隐藏和显示 HTML 元素。
- AngularJS 可以在 HTML 元素"背后"添加代码。
- AngularJS 支持输入验证。

AngularJS 指令

正如您所看到的，AngularJS 指令是以 **ng** 作为前缀的 HTML 属性。

ng-init 指令初始化 AngularJS 应用程序变量。

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John'">

<p>姓名为 <span ng-bind="firstName"></span></p>

</div>
```

>  HTML5 允许扩展的（自制的）属性，以 **data-** 开头。
> AngularJS 属性以 **ng-** 开头，但是您可以使用 **data-ng-** 来让网页对 HTML5 有效。

带有有效的 HTML5：

AngularJS 实例

```
<div data-ng-app="" data-ng-init="firstName='John'">

<p>姓名为 <span data-ng-bind="firstName"></span></p>

</div>
```

AngularJS 表达式

AngularJS 表达式写在双大括号内：**。

AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。

AngularJS 将在表达式书写的位置"输出"数据。

AngularJS 表达式 很像 **JavaScript** 表达式：它们可以包含文字、运算符和变量。

实例 10 或 undefined undefined

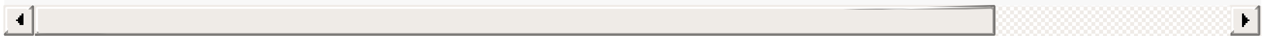
AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="">
  <p>我的第一个表达式： {{ 5 + 5 }}</p>
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

</body>
</html>
```



AngularJS 表达式

AngularJS 使用 表达式 把数据绑定到 HTML。

AngularJS 表达式

AngularJS 表达式写在双大括号内：**。

AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。

AngularJS 将在表达式书写的位置"输出"数据。

AngularJS 表达式 很像 **JavaScript** 表达式：它们可以包含文字、运算符和变量。

实例 10 或 undefined undefined

AngularJS 数字

AngularJS 数据就像 JavaScript 数字：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>总价： {{ quantity * cost }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;cost=5">

<p>总价： <span ng-bind="quantity * cost"></span></p>

</div>
```

>  使用 **ng-init** 不是很常见。您将在控制器一章中学习到一个更好的初始化数据的方式。

AngularJS 字符串

AngularJS 字符串就像 JavaScript 字符串：

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>姓名： {{ firstName + " " + lastName }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John';lastName='Doe'">

<p>姓名： <span ng-bind="firstName + ' ' + lastName"></span></p>

</div>
```

AngularJS 对象

AngularJS 对象就像 JavaScript 对象：

AngularJS 实例

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>姓为 {{ person.lastName }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="person={firstName:'John',lastName:'Doe'}">

<p>姓为 <span ng-bind="person.lastName"></span></p>

</div>
```

AngularJS 数组

AngularJS 数组就像 JavaScript 数组：

AngularJS 实例

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>第三个值为 {{ points[2] }}</p>

</div>
```

使用 ng-bind 的相同实例：

AngularJS 实例

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">

<p>第三个值为 <span ng-bind="points[2]"></span></p>

</div>
```

AngularJS 指令

AngularJS 通过被称为 指令 的新属性来扩展 HTML。

AngularJS 指令

AngularJS 指令是扩展的 HTML 属性，带有前缀 **ng-**。

ng-app 指令初始化一个 AngularJS 应用程序。

ng-init 指令初始化应用程序数据。

ng-model 指令把应用程序数据绑定到 HTML 元素。

AngularJS 实例

```
<div ng-app="" ng-init="firstName='John'">

  <p>在输入框中尝试输入：</p>
  <p>姓名：<input type="text" ng-model="firstName"></p>
  <p>你输入的为： {{ firstName }}</p>

</div>
```

ng-app 指令告诉 AngularJS，<div> 元素是 AngularJS 应用程序 的"所有者"。

 一个网页可以包含多个运行在不同元素中的 AngularJS 应用程序。

数据绑定

上面实例中的 ** 表达式是一个 AngularJS 数据绑定表达式。

AngularJS 中的数据绑定，同步了 AngularJS 表达式与 AngularJS 数据。

** 是通过 ng-model="firstName"** 进行同步。

在下一个实例中，两个文本域是通过两个 ng-model 指令同步的：

AngularJS 实例

```
<div ng-app="" ng-init="quantity=1;price=5">

<h2>价格计算器</h2>

数量： <input type="number" ng-model="quantity">
价格： <input type="number" ng-model="price">

<p><b>总价：</b> {{ quantity * price }}</p>

</div>
```

 使用 **ng-init** 不是很常见。您将在控制器一章中学习到一个更好的初始化数据的方式。

重复 HTML 元素

ng-repeat 指令会重复一个 HTML 元素：

AngularJS 实例

```
<div ng-app="" ng-init="names=['Jani','Hege','Kai']">
  <p>使用 ng-repeat 来循环数组</p>
  <ul>
    <li ng-repeat="x in names">
      {{ x }}
    </li>
  </ul>
<div>
```

ng-repeat 指令用在一个对象数组上：

AngularJS 实例

```
<div ng-app="" ng-init="names=[
{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">

<p>循环对象：</p>
<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>
```

 AngularJS 完美支持数据库的 CRUD（增加Create、读取Read、更新Update、删除Delete）应用程序。
把实例中的对象想象成数据库中的记录。

ng-app 指令

ng-app 指令定义了 AngularJS 应用程序的 根元素。

ng-app 指令在网页加载完毕时会自动引导（自动初始化）应用程序。

稍后您将学习到 **ng-app** 如何通过一个值（比如 ng-app="myModule"）连接到代码模块。

ng-init 指令

ng-init 指令为 AngularJS 应用程序定义了 初始值。

通常情况下，不使用 ng-init。您将使用一个控制器或模块来代替它。

稍后您将学习更多有关控制器和模块的知识。

ng-model 指令

ng-model 指令 绑定 **HTML** 元素 到应用程序数据。

ng-model 指令也可以：

- 为应用程序数据提供类型验证（number、email、required）。
- 为应用程序数据提供状态（invalid、dirty、touched、error）。
- 为 HTML 元素提供 CSS 类。
- 绑定 HTML 元素到 HTML 表单。

ng-repeat 指令

ng-repeat 指令对于集合中（数组中）的每个项会 克隆一次 **HTML** 元素。

AngularJS 控制器

AngularJS 控制器 控制 AngularJS 应用程序的数据。

AngularJS 控制器是常规的 **JavaScript** 对象。

AngularJS 控制器

AngularJS 应用程序被控制器控制。

ng-controller 指令定义了应用程序控制器。

控制器是 **JavaScript** 对象，由标准的 JavaScript 对象的构造函数 创建。

控制器的 **$scope** 是控制器所指向的应用程序/HTML 元素。

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{person.firstName + " " + person.lastName}}

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe"
    };
}
</script>
```

实例讲解：

AngularJS 应用程序由 **ng-app** 定义。应用程序在 <div> 内运行。

ng-controller 指令把控制器命名为 **object**。

函数 **personController** 是一个标准的 JavaScript 对象的构造函数。

控制器对象有一个属性：**$scope.person**。

person 对象有两个属性：**firstName** 和 **lastName**。

ng-model 指令绑定输入域到控制器的属性（firstName 和 lastName）。

控制器属性

上面的实例演示了一个带有 lastName 和 firstName 这两个属性的控制器对象。

控制器也可以把函数作为对象属性：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{person.fullName()}}

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe",
        fullName: function() {
            var x;
            x = $scope.person;
            return x.firstName + " " + x.lastName;
        }
    };
}
</script>
```

控制器方法

控制器也可以带有方法：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{fullName()}}

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe",
     };
     $scope.fullName = function() {
         var x;
         x = $scope.person;
         return x.firstName + " " + x.lastName;
     };
}
</script>
```

外部文件中的控制器

在大型的应用程序中，通常是把控制器存储在外部文件中。

只需要把 <script> 标签中的代码复制到名为 personController.js 的外部文件中即可：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

名： <input type="text" ng-model="person.firstName"><br>
姓： <input type="text" ng-model="person.lastName"><br>
<br>
姓名： {{person.firstName + " " + person.lastName}}

</div>

<script src="personController.js"></script>
```

另一个实例

下面的实例我们将创建一个新的控制器文件：

```
function namesController($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Hege',country:'Sweden'},
        {name:'Kai',country:'Denmark'}
    ];
}
```

然后在应用程序中使用这个控制器文件：

AngularJS 实例

```
<div ng-app="" ng-controller="namesController">

<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

</div>

<script src="namesController.js"></script>
```

AngularJS 过滤器

过滤器可以使用一个管道字符（|）添加到表达式和指令中。

AngularJS 过滤器

AngularJS 过滤器可用于转换数据：

过滤器	描述
currency	格式化数字为货币格式。
filter	从数组项中选择一个子集。
lowercase	格式化字符串为小写。
orderBy	根据某个表达式排列数组。
uppercase	格式化字符串为大写。

向表达式添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到表达式中。

（下面的两个实例，我们将使用前面章节中提到的 person 控制器）

uppercase 过滤器格式化字符串为大写：

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

<p>姓名为 {{ person.lastName | uppercase }}</p>

</div>
```

lowercase 过滤器格式化字符串为小写：

AngularJS 实例

```
<div ng-app="" ng-controller="personController"">

<p>姓名为 {{ person.lastName | lowercase }}</p>

</div>
```

currency 过滤器

currency 过滤器格式化数字为货币格式：

AngularJS 实例

```
<div ng-app="" ng-controller="costController">

数量：<input type="number" ng-model="quantity">
价格：<input type="number" ng-model="price">

<p>总价 = {{ (quantity * price) | currency }}</p>

</div>
```

向指令添加过滤器

过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中。

orderBy 过滤器根据某个表达式排列数组：

AngularJS 实例

```
<div ng-app="" ng-controller="namesController">

<p>循环对象：</p>
<ul>
  <li ng-repeat="x in names | orderBy:'country'">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>

<div>
```

过滤输入

输入过滤器可以通过一个管道字符（|）和一个过滤器添加到指令中，该过滤器后跟一个冒号和一个模型名称。

filter 过滤器从数组中选择一个子集：

AngularJS 实例

```
<div ng-app="" ng-controller="namesController">

<p>输入过滤：</p>
<p><input type="text" ng-model="name"></p>

<ul>
  <li ng-repeat="x in names | filter:name | orderBy:'country'">
    {{ (x.name | uppercase) + ', ' + x.country }}
  </li>
</ul>

</div>
```

AngularJS XMLHttpRequest

$http 是 AngularJS 中的一个核心服务，用于读取远程服务器的数据。

读取 JSON 文件

以下是存储在web服务器上的 JSON 文件：

http://www.w3cschool.cc/try/angularjs/data/Customers_JSON.php

```
[
{
"Name" : "Alfreds Futterkiste",
"City" : "Berlin",
"Country" : "Germany"
},
{
"Name" : "Berglunds snabbk?p",
"City" : "Lule?",
"Country" : "Sweden"
},
{
"Name" : "Centro comercial Moctezuma",
"City" : "México D.F.",
"Country" : "Mexico"
},
{
"Name" : "Ernst Handel",
"City" : "Graz",
"Country" : "Austria"
},
{
"Name" : "FISSA Fabrica Inter. Salchichas S.A.",
"City" : "Madrid",
"Country" : "Spain"
},
{
"Name" : "Galería del gastrónomo",
"City" : "Barcelona",
"Country" : "Spain"
},
{
"Name" : "Island Trading",
"City" : "Cowes",
"Country" : "UK"
},
{
"Name" : "K?niglich Essen",
```

```
"City" : "Brandenburg",
"Country" : "Germany"
},
{
"Name" : "Laughing Bacchus Wine Cellars",
"City" : "Vancouver",
"Country" : "Canada"
},
{
"Name" : "Magazzini Alimentari Riuniti",
"City" : "Bergamo",
"Country" : "Italy"
},
{
"Name" : "North/South",
"City" : "London",
"Country" : "UK"
},
{
"Name" : "Paris spécialités",
"City" : "Paris",
"Country" : "France"
},
{
"Name" : "Rattlesnake Canyon Grocery",
"City" : "Albuquerque",
"Country" : "USA"
},
{
"Name" : "Simons bistro",
"City" : "K?benhavn",
"Country" : "Denmark"
},
{
"Name" : "The Big Cheese",
"City" : "Portland",
"Country" : "USA"
},
{
"Name" : "Vaffeljernet",
"City" : "?rhus",
"Country" : "Denmark"
},
{
"Name" : "Wolski Zajazd",
"City" : "Warszawa",
"Country" : "Poland"
}
]
```

AngularJS $http

AngularJS $http 是一个用于读取web服务器上数据的服务。

$http.get(url) 是用于读取服务器数据的函数。

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<ul>
  <li ng-repeat="x in names">
    {{ x.Name + ', ' + x.Country }}
  </li>
</ul>

</div>

<script>
function customersController($scope,$http) {
    $http.get("http://www.w3cschool.cc/try/angularjs/data/Customers
    .success(function(response) {$scope.names = response;});
}
</script>
```

应用解析:

AngularJS 应用通过 **ng-app** 定义。应用在 <div> 中执行。

ng-controller 指令设置了 **controller** 对象 名。

函数 **customersController** 是一个标准的 JavaScript 对象构造器。

控制器对象有一个属性: **$scope.names**。

$http.get() 从web服务器上读取静态 **JSON** 数据。

服务器数据文件为：**http://www.w3cschool.cc/try/angularjs/data/Customers_JSON.php**。

当从服务端载入 JSON 数据时，**$scope.names** 变为一个数组。

 以上代码也可以用于读取数据库数据。

AngularJS 表格

ng-repeat 指令可以完美的显示表格。

在表格中显示数据

使用 angular 显示表格是非常简单的：

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
function customersController($scope,$http) {
  $http.get("http://www.w3cschool.cc/try/angularjs/data/Customers_J
  .success(function(response) {$scope.names = response;});
}
</script>
```

使用 CSS 样式

为了让页面更加美观，我们可以在页面中使用CSS：

CSS 样式

```
<style>
table, th , td {
  border: 1px solid grey;
  border-collapse: collapse;
  padding: 5px;
}
table tr:nth-child(odd) {
  background-color: #f1f1f1;
}
table tr:nth-child(even) {
  background-color: #ffffff;
}
</style>
```

排序显示

如果需要对表格进行排序，我们可以添加 **orderBy** 过滤器:

AngularJS 实例

```
<table>
  <tr ng-repeat="x in names | orderBy : 'Country'">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>
```

Try it Yourself ?

使用 uppercase 滤器

如果字母要转换为大写，可以添加 **uppercase** 过滤器:

AngularJS 实例

```
<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country | uppercase}}</td>
  </tr>
</table>
```

AngularJS SQL

在前面章节中的代码也可以用于读取数据库中的数据。

使用 PHP 从 MySQL 中获取数据

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
function customersController($scope,$http) {
    var site = "http://www.w3cschool.cc";
    var page = "/try/angularjs/data/Customers_MySQL.php";
    $http.get(site + page)
    .success(function(response) {$scope.names = response;});
}
</script>
```

ASP.NET 中执行 SQL 获取数据

AngularJS 实例

```
<div ng-app="" ng-controller="customersController">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
function customersController($scope,$http) {
    var site = "http://www.w3cschool.cc";
    var page = "/try/angularjs/data/Customers_SQL.aspx";
    $http.get(site + page)
    .success(function(response) {$scope.names = response;});
}
</script>
```

PHP 读取 MySQL 数据代码

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: text/html; charset=UTF-8");

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind")

$result = $conn->query("SELECT CompanyName, City, Country FROM Cust

$outp "[";
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
    if ($outp != "[") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'   . $rs["City"]        . '",';
    $outp .= '"Country":"'. $rs["Country"]     . '"}';
}
$outp .="]";

$conn->close();

echo($outp);
?>
```

PHP 读取 MS Access 代码

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: text/html; charset=ISO-8859-1");

$conn = new COM("ADODB.Connection");
$conn->open("PROVIDER=Microsoft.Jet.OLEDB.4.0;Data Source=Northwind

$rs = $conn->execute("SELECT CompanyName, City, Country FROM Custom

$outp = "[";
while (!$rs->EOF) {
    if ($outp != "[") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'   . $rs["City"]        . '",';
    $outp .= '"Country":"'. $rs["Country"]     . '"}';
    $rs->MoveNext();
}
$outp .= "]";

$conn->close();

echo ($outp);
?>
```

服务端 ASP.NET, VB 和 MS Access 代码

```
<%@ Import Namespace="System.IO"%>
<%@ Import Namespace="System.Data"%>
<%@ Import Namespace="System.Data.OleDb"%>
<%
Response.AppendHeader("Access-Control-Allow-Origin", "*")
Dim conn As OleDbConnection
Dim objAdapter As OleDbDataAdapter
Dim objTable As DataTable
Dim objRow As DataRow
Dim objDataSet As New DataSet()
Dim outp
Dim c
conn = New OledbConnection("Provider=Microsoft.Jet.OLEDB.4.0;data s
objAdapter = New OledbDataAdapter("SELECT CompanyName, City, Countr
objAdapter.Fill(objDataSet, "myTable")
objTable=objDataSet.Tables("myTable")

outp = "["
c = chr(34)
for each x in objTable.Rows
if outp <> "[" then outp = outp & ","
outp = outp & "{" & c & "Name"    & c & ":" & c & x("CompanyName")
outp = outp &       c & "City"    & c & ":" & c & x("City")
outp = outp &       c & "Country" & c & ":" & c & x("Country")
next

outp = outp & "]"
response.write(outp)
conn.close
%>
```

服务端 ASP.NET, VB Razor 和 SQL Lite 代码

```
@{
Response.AppendHeader("Access-Control-Allow-Origin", "*")
var db = Database.Open("Northwind");
var query = db.Query("SELECT CompanyName, City, Country FROM Custor
var outp ="["
}
@foreach(var row in query)
{
if outp <> "[" then outp = outp + ","
outp = outp + "{" + c + "Name"    + c + ":" + c + @row.CompanyName
outp = outp +       c + "City"    + c + ":" + c + @row.City
outp = outp +       c + "Country" + c + ":" + c + @row.Country
}
outp = outp + "]"
@outp
```

AngularJS HTML DOM

AngularJS 有自己的 HTML 属性指令。

ng-disabled 指令

ng-disabled 指令直接绑定应用程序数据到 HTML 的 disabled 属性。

AngularJS 实例

```
<div ng-app="">

<p>
<button ng-disabled="mySwitch">点我！</button>
</p>

<p>
<input type="checkbox" ng-model="mySwitch">按钮
</p>

</div>
```

实例讲解：

ng-disabled 指令绑定应用程序数据 "mySwitch" 到 HTML 的 disabled 属性。

ng-model 指令绑定 "mySwitch" 到 HTML input checkbox 元素的内容（value）。

ng-show 指令

ng-show 指令隐藏或显示一个 HTML 元素。

AngularJS 实例

```
<div ng-app="">

<p ng-show="true">我是可见的。</p>

<p ng-show="false">我是不可见的。</p>

</div>
```

您可以使用一个评估为 true or false 的表达式（比如 ng-show="hour < 12"）来隐藏和显示 HTML 元素。

在下一章中，有另一个实例，通过单击一个按钮来隐藏一个 HTML 元素。

AngularJS HTML 事件

AngularJS 有自己的 HTML 事件指令。

ng-click 指令

ng-click 指令定义了一个 AngularJS 单击事件。

AngularJS 实例

```
<div ng-app="" ng-controller="myController">

<button ng-click="count = count + 1">点我！</button>

<p>{{ count }}</p>

</div>
```

隐藏 HTML 元素

ng-hide 指令用于设置应用的一部分 不可见 。

ng-hide="true" 让 HTML 元素 不可见。

ng-hide="false" 让元素可见。

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

<button ng-click="toggle()">隐藏/显示</button>

<p ng-hide="myVar">
名: <input type="text" ng-model="person.firstName"><br>
姓: <input type="text" ng-model="person.lastName"><br>
<br>
姓名: {{person.firstName + " " + person.lastName}}
</p>

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe"
    };
    $scope.myVar = false;
    $scope.toggle = function() {
        $scope.myVar = !$scope.myVar;
    };
}
</script>
```

应用解析:

personController的第一部分与控制器章节类似。

应用有一个默认属性: **$scope.myVar = false;**

ng-hide 指令设置应用中的元素不可见。

toggle() 函数用于切换 **myVar** 变量的值（true 和 false）。

ng-hide="true" 让元素 不可见。

显示 HTML 元素

ng-show 指令可用于设置应用中的一部分可见 。

ng-show="false" 可以设置 HTML 元素 不可见。

ng-show="true" 可以以设置 HTML 元素可见。

以下实例使用了 ng-show 指令:

AngularJS 实例

```
<div ng-app="" ng-controller="personController">

<button ng-click="toggle()">隐藏/显示</button>

<p ng-show="myVar">
名: <input type="text" ng-model="person.firstName"><br>
姓: <input type="text" ng-model="person.lastName"><br>
<br>
姓名: {{person.firstName + " " + person.lastName}}
</p>

</div>

<script>
function personController($scope) {
    $scope.person = {
        firstName: "John",
        lastName: "Doe"
    };
    $scope.myVar = true;
    $scope.toggle = function() {
        $scope.myVar = !$scope.myVar;
    };
}
</script>
```

AngularJS 模块

模块定义了您的应用程序。

所有的控制器都应该属于一个模块。

模块保持全局命名空间中的整洁。

AngularJS 模块实例

在本实例中，"myApp.js" 包含了一个应用程序模块定义，"myCtrl.js" 包含了一个控制器：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

控制器污染了全局命名空间

本教程中，截至目前为止的所有实例都使用了全局函数。

在所有的应用程序中，都应该尽量避免使用全局变量和全局函数。

全局值（变量或函数）可被其他脚本重写或破坏。

为了解决这个问题，AngularJS 使用了模块。

AngularJS 模块

使用一个简单的 控制器：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>
function myCtrl($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
}
</script>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

</body>
</html>
```

使用一个由 模块 替代的控制器：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<head>
<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'
</head>

<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>
var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>

</body>
</html>
```



请注意，本实例中，AngularJS 库是在 <head> 区域被加载。

模块定义应放置在何处？

对于 HTML 应用程序，通常建议把所有的脚本都放置在 <body> 元素的最底部。

这会提高网页加载速度，因为 HTML 加载不受制于脚本加载。

在上面的多个 AngularJS 实例中，您将看到 AngularJS 库是在文档的 <head> 区域被加载。

在上面的实例中，AngularJS 在 <head> 元素中被加载，因为对 angular.module 的调用只能在库加载完成后才能进行。

另一个解决方案是在 <body> 元素中加载 AngularJS 库，但是必须放置在您的 AngularJS 脚本前面：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

<script>
var app = angular.module("myApp", []);

app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>

</body>
</html>
```

AngularJS 应用程序文件

现在您已经知道模块是什么以及它们是如何工作的，现在您可以尝试创建您自己的应用程序文件。

您的应用程序至少应该有一个模块文件，一个控制器文件。

首先，创建模块文件 "myApp.js"：

```
var app = angular.module("myApp", []);
```

然后，创建控制器文件。本实例中是 "myCtrl.js"：

```
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
```

最后，编辑您的 HTML 页面：

AngularJS 实例

```
<!DOCTYPE html>
<html>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="//www.w3cschool.cc/try/angularjs/1.2.5/angular.min.js'

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>

</body>
</html>
```

AngularJS 表单

AngularJS 表单是输入控件的集合。

HTML 控件

以下 HTML input 元素被称为 HTML 控件:

- input 元素
- select 元素
- button 元素
- textarea 元素

HTML 表单

HTML 表单通常与 HTML 控件同时存在。

AngularJS 表单实例

```
<script>function ExampleController($scope) { $scope.master = {"firs

<form novalidate="" class="ng-pristine ng-valid">First Name:
<input type="text" ng-model="user.firstName" class="ng-pristine ng-
Last Name:
<input type="text" ng-model="user.lastName" class="ng-pristine ng-v

<button ng-click="reset()">RESET</button></form>

form = {"firstName":"John","lastName":"Doe"}

master = {"firstName":"John","lastName":"Doe"}
```

应用程序代码

```
<div ng-app="" ng-controller="formController">
  <form novalidate>
    First Name:<br>
    <input type="text" ng-model="user.firstName"><br>
    Last Name:<br>
    <input type="text" ng-model="user.lastName">
    <br><br>
    <button ng-click="reset()">RESET</button>
  </form>
  <p>form = {{user}}</p>
  <p>master = {{master}}</p>
</div>

<script>
function formController ($scope) {
    $scope.master = {firstName: "John", lastName: "Doe"};
    $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
    };
    $scope.reset();
};
</script>
```

 HTML 属性 **novalidate** 用于禁用浏览器的默认验证。

实例解析

AngularJS **ng-model** 指令用于绑定 input 元素到模型中。

模型对象 **master** 的值为 {"firstName" : "John", "lastName" : "Doe"}。

模型函数 **reset** 设置了模型对象 **user** 等于 master。

AngularJS 输入验证

AngularJS 表单和控件可以验证输入的数据。

输入验证

在前面的几个章节中，你已经学到关于 AngularJS 表单和控件的知识。

AngularJS 表单和控件可以提供验证功能，并对用户输入的非法数据进行警告。

>  客户端的验证不能确保用户输入数据的安全，所以服务端的数据验证也是必须的。

应用代码

```
<!DOCTYPE html>
<html>

<body>
<h2>Validation Example</h2>

<form  ng-app=""  ng-controller="validateCtrl"
name="myForm" novalidate>

<p>Username:<br>
  <input type="text" name="user" ng-model="user" required>
  <span style="color:red" ng-show="myForm.user.$dirty && myForm.use
  <span ng-show="myForm.user.$error.required">Username is required
  </span>
</p>

<p>Email:<br>
  <input type="email" name="email" ng-model="email" required>
  <span style="color:red" ng-show="myForm.email.$dirty && myForm.en
  <span ng-show="myForm.email.$error.required">Email is required.</
  <span ng-show="myForm.email.$error.email">Invalid email address.<
  </span>
</p>

<p>
  <input type="submit"
  ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||
  myForm.email.$dirty && myForm.email.$invalid">
</p>

</form>

<script src="//apps.bdimg.com/libs/angular.js/1.2.15/angular.min.js
<script>
function validateCtrl($scope) {
    $scope.user = 'John Doe';
    $scope.email = 'john.doe@gmail.com';
}
</script>

</body>
</html>
```

 HTML 表单属性 **novalidate** 用于禁用浏览器默认的验证。

实例解析

AngularJS **ng-model** 指令用于绑定输入元素到模型中。

模型对象有两个属性： **user** 和 **email**。

我们使用了 **ng-show**指令， color:red 在邮件是 **$dirty** 或 **$invalid** 才显示。

AngularJS Bootstrap

AngularJS 的首选样式表是 Twitter Bootstrap， Twitter Bootstrap 是目前最受欢迎的前端框架。

查看 Bootstrap教程。

Bootstrap

你可以在你的 AngularJS 应用中加入 Twitter Bootstrap，你可以在你的 <head>元素中添加如下代码:

```
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3
```

如果站点在国内，建议使用百度静态资源库的Bootstrap，代码如下：

```
<link rel="stylesheet" href="//apps.bdimg.com/libs/bootstrap/3.2.0/
```

以下是一个完整的 HTML 实例, 使用了 AngularJS 指令和 Bootstrap 类。

HTML 代码

```
<!DOCTYPE html>
<html ang-app="">
<head>
<link rel="stylesheet" href="http://apps.bdimg.com/libs/bootstrap/3
</head>

<body ng-controller="userController">
<div class="container">

<h3>Users</h3>

<table class="table table-striped">
  <thead><tr>
    <th>Edit</th>
    <th>First Name</th>
    <th>Last Name</th>
  </tr></thead>
  <tbody><tr ng-repeat="user in users">
    <td>
```

```
      <button class="btn" ng-click="editUser(user.id)">
      <span class="glyphicon glyphicon-pencil"></span>  E
      </button>
    </td>
    <td>{{ user.fName }}</td>
    <td>{{ user.lName }}</td>
  </tr></tbody>
</table>

<hr>
<button class="btn btn-success" ng-click="editUser('new')">
  <span class="glyphicon glyphicon-user"></span> Create New User
</button>
<hr>

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<div class="form-group">
  <label class="col-sm-2 control-label">First Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="fName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Last Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="lName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Password:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw1" placeholder="Password"
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Repeat:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw2" placeholder="Repeat Pa
  </div>
</div>
</form>

<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
</div>

<script src = "http://apps.bdimg.com/libs/angular.js/1.2.15/angular
<script src = "myUsers.js"></script>
```

```
</body>
</html>>
```

指令解析

AngularJS 指令	描述
html ng-app	为 `<html>` 元素定义一个应用(未命名)
body ng-controller	为 `<body>` 元素定义一个控制器
tr ng-repeat	循环 users 对象数组，每个 user 对象放在 `<tr>` 元素中。
button ng-click	当点击 `<button>` 元素时调用函数 editUser()
h3 ng-show	如果 edit = true 显示 `<h3>` 元素
h3 ng-hide	如果 edit = true 隐藏 `<h3>` 元素
input ng-model	为应用程序绑定 `<input>` 元素
button ng-disabled	如果发生错误或者 ncomplete = true 禁用 `<button>` 元素

Bootstrap 类解析

元素	**Bootstrap 类**	定义
`<div>`	container	内容容器
`<table>`	table	表格
`<table>`	table-striped	带条纹背景的表格
`<button>`	btn	按钮
`<button>`	btn-success	成功按钮
`<span>`	glyphicon	字形图标
`<span>`	glyphicon-pencil	铅笔图标
`<span>`	glyphicon-user	用户图标
`<span>`	glyphicon-save	保存图标
`<form>`	form-horizontal	水平表格
`<div>`	form-group	表单组
`<label>`	control-label	控制器标签
`<label>`	col-sm-2	跨越 2 列
`<div>`	col-sm-10	跨越 10 列

JavaScript 代码

```
function userController($scope) {
$scope.fName = '';
$scope.lName = '';
$scope.passw1 = '';
$scope.passw2 = '';
$scope.users = [
{id:1, fName:'Hege',  lName:"Pege" },
{id:2, fName:'Kim',   lName:"Pim" },
{id:3, fName:'Sal',   lName:"Smith" },
{id:4, fName:'Jack',  lName:"Jones" },
{id:5, fName:'John',  lName:"Doe" },
{id:6, fName:'Peter', lName:"Pan" }
];
$scope.edit = true;
$scope.error = false;
$scope.incomplete = false;

$scope.editUser = function(id) {
  if (id == 'new') {
    $scope.edit = true;
    $scope.incomplete = true;
    $scope.fName = '';
    $scope.lName = '';
    } else {
    $scope.edit = false;
    $scope.fName = $scope.users[id-1].fName;
    $scope.lName = $scope.users[id-1].lName;
  }
};

$scope.$watch('passw1',function() {$scope.test();});
$scope.$watch('passw2',function() {$scope.test();});
$scope.$watch('fName', function() {$scope.test();});
$scope.$watch('lName', function() {$scope.test();});

$scope.test = function() {
  if ($scope.passw1 !== $scope.passw2) {
    $scope.error = true;
    } else {
    $scope.error = false;
  }
  $scope.incomplete = false;
  if ($scope.edit && (!$scope.fName.length ||
  !$scope.lName.length ||
  !$scope.passw1.length || !$scope.passw2.length)) {
       $scope.incomplete = true;
  }
};

}
```

JavaScript 代码解析

Scope 属性	用途
$scope.fName	模型变量 (用户名)
$scope.lName	模型变量 (用户姓)
$scope.passw1	模型变量 (用户密码 1)
$scope.passw2	模型变量 (用户密码 2)
$scope.users	模型变量 (用户的数组)
$scope.edit	当用户点击创建用户时设置为true。
$scope.error	如果 passw1 不等于 passw2 设置为 true
$scope.incomplete	如果每个字段都为空(length = 0)设置为 true
$scope.editUser	设置模型变量
$scope.watch	监控模型变量
$scope.test	验证模型变量的错误和完整性

AngularJS Include（包含）

使用 AngularJS, 你可以在 HTML 中包含 HTML 文件。

在未来的HTML中包含 HTML 文件

在 HTML 中，目前还不支持包含 HTML 文件的功能。

W3C 已经建议 http://www.w3.org 在未来的 HTML 中支持包含HTML的功能，格式如下:

```
<link rel="import" href="/path/navigation.html">
```

服务端包含

大部分web服务器支持服务端脚本的包含 (**SSI：Server Side Includes**)。

使用 SSI, 你可以在HTML页面发送至浏览器前包含 HTML。

PHP 实例

```
<?php require("navigation.php"); ?>
```

客户端包含

客户端在 HTML 中使用 JavaScript 有多种方式可以包含 HTML 文件。

通常我们使用 http 请求 (**AJAX**) 从服务端获取数据，返回的数据我们可以通过 使用 **innerHTML** 写入到 HTML 元素中。

AngularJS 包含

使用 AngularJS, 你可以使用 **ng-include** 指令来包含 HTML 内容:

实例

```
<body>

<div class="container">
  <div ng-include="'myUsers_List.htm'"></div>
  <div ng-include="'myUsers_Form.htm'"></div>
</div>

</body>
```

步骤如下。

步骤 1: 创建 HTML 列表

myUsers_List.html

```
<table class="table table-striped">
  <thead><tr>
    <th>Edit</th>
    <th>First Name</th>
    <th>Last Name</th>
  </tr></thead>
  <tbody><tr ng-repeat="user in users">
    <td>
      <button class="btn" ng-click="editUser(user.id)">
        <span class="glyphicon glyphicon-pencil"></span>  
      </button>
    </td>
    <td>{{ user.fName }}</td>
    <td>{{ user.lName }}</td>
  </tr></tbody>
</table>
```

步骤 2: 创建 HTML 表单

myUsers_List.html

```
<button class="btn btn-success" ng-click="editUser('new')">
  <span class="glyphicon glyphicon-user"></span> Create New User
</button>
<hr>

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<div class="form-group">
  <label class="col-sm-2 control-label">First Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="fName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Last Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="lName" ng-disabled="!edit" placeho
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Password:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw1" placeholder="Password'
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Repeat:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw2" placeholder="Repeat Pa
  </div>
</div>
</form>

<hr>
<button class="btn btn-success" ng-disabled="error || incomplete">
  <span class="glyphicon glyphicon-save"></span> Save Changes
</button>
```

步骤 3: 创建主页

myUsers.html

```
<!DOCTYPE html>
<html ng-app="">
<head>
<link rel="stylesheet" href = "http://apps.bdimg.com/libs/bootstrap
</head>

<body ng-controller="userController">

<div class="container">
<div ng-include="'myUsers_List.htm'"></div>
<div ng-include="'myUsers_Form.htm'"></div>
</div>

<script src= "http://apps.bdimg.com/libs/angular.js/1.2.15/angular.
<script src= "myUsers.js"></script>

</body>
</html>
```

AngularJS 应用程序

现在是时候创建一个真正的 AngularJS 应用程序了。

AngularJS 应用程序

您已经学习了足够多关于 AngularJS 的知识，现在可以开始创建您的第一个 AngularJS 应用程序：

我的笔记

```
<textarea cols="40" rows="10"></textarea>

<button>保存</button> <button>清除</button>

剩下的字符数：
```

应用程序讲解

AngularJS 实例

```
<div ng-app="myTodoApp" ng-controller="myTodoCtrl">

<h2>我的笔记</h2>

<p><textarea ng-model="message" cols="40" rows="10"></textarea></p>

<p>
<button ng-click="save()">保存</button>
<button ng-click="clear()">清除</button>
</p>

<p>剩下的字符数：<span ng-bind="left()"></span></p>

</div>

<script src="myTodoApp.js"></script>
<script src="myTodoCtrl.js"></script>
```

应用程序文件 "myTodoApp.js"：

```
var app = angular.module("myTodoApp", []);
```

控制器文件 "myTodoCtrl.js"：

```
app.controller("myTodoCtrl", function($scope) {
    $scope.message = "";
    $scope.left  = function() {return 100 - $scope.message.length;]
    $scope.clear = function() {$scope.message="";};
    $scope.save  = function() {$scope.message="";};
});
```

HTML 页面中的一个 <div>，指向 ng-app="**myTodoApp**" 和 ng-controller="**myTodoCtrl**"：

```
<div ng-app="myTodoApp" ng-controller="myTodoCtrl">
```

一个 **ng-model** 指令，绑定一个 <textarea> 到控制器变量 **message**：

```
<textarea ng-model="message" cols="40" rows="10"></textarea>
```

两个 **ng-click** 事件，调用控制器函数 **clear()** 和 **save()**：

```
<button ng-click="save()">保存</button>
<button ng-click="clear()">清除</button>
```

一个 **ng-bind** 指令，绑定控制器函数 **left()** 到一个 <span>，字符会向左对齐显示：

```
剩下的字符数：<span ng-bind="left()"></span>
```

两个应用程序库被添加到 HTML 页面：

```
<script src="myTodoApp.js"></script>
<script src="myTodoCtrl.js"></script>
```

AngularJS 参考手册

AngularJS 指令

本教程中使用的 AngularJS 指令：

指令	描述	讲解
ng_app	定义应用程序的根元素。	指令
ng_bind	绑定 HTML 元素到应用程序数据。	简介
ng_click	定义元素被单击时的行为。	HTML 事件
ng_controller	为应用程序定义控制器对象。	控制器
ng_disabled	绑定应用程序数据到 HTML 的 disabled 属性。	HTML DOM
ng_init	为应用程序定义初始值。	指令
ng_model	绑定应用程序数据到 HTML 元素。	指令
ng_repeat	为控制器中的每个数据定义一个模板。	指令
ng_show	显示或隐藏 HTML 元素。	HTML DOM

AngularJS 过滤器

本教程中使用的 AngularJS 过滤器：

过滤器	描述
currency	格式化数字为货币格式。
filter	从数组项中选择一个子集。
lowercase	格式化字符串为小写。
orderBy	根据某个表达式排列数组。
uppercase	格式化字符串为大写。

有关过滤器的具体知识在 AngularJS 过滤器 一章中进行讲解。

免责声明

W3School提供的内容仅用于培训。我们不保证内容的正确性。通过使用本站内容随之而来的风险与本站无关。W3School简体中文版的所有内容仅供测试，对任何法律问题及风险不承担任何责任。